# Process...Modeling...EA Repository Development

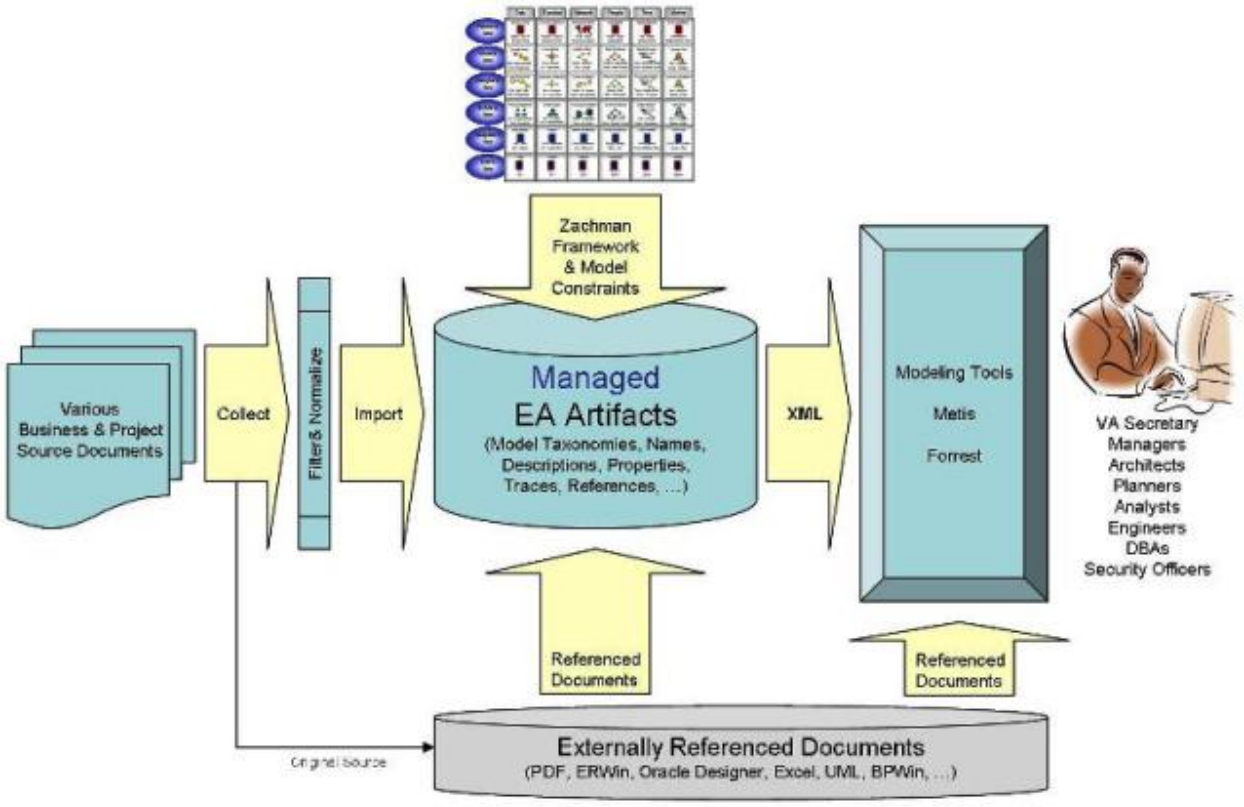
## Table of contents


1 EA Repository Development ..... 2
1.1 EA Repository Introduction ..... 2
1.2 EA Information Capture, Management, and Delivery ..... 3
1.3 CaliberRM Mechanisms ..... 4
1.3.1 Basic Mechanisms ..... 5
1.3.2 Utilities ..... 6
1.3.3 Built-in Properties ..... 7
1.3.4 User-Extended Properties ..... 7
1.3.5 Data Management ..... 8
1.4 Referencing External Mandates ..... 8

# 1. EA Repository Development

## 1.1. EA Repository Introduction

The following graphic illustrates the central role the EA repository plays in the VA Enterprise Architecture.



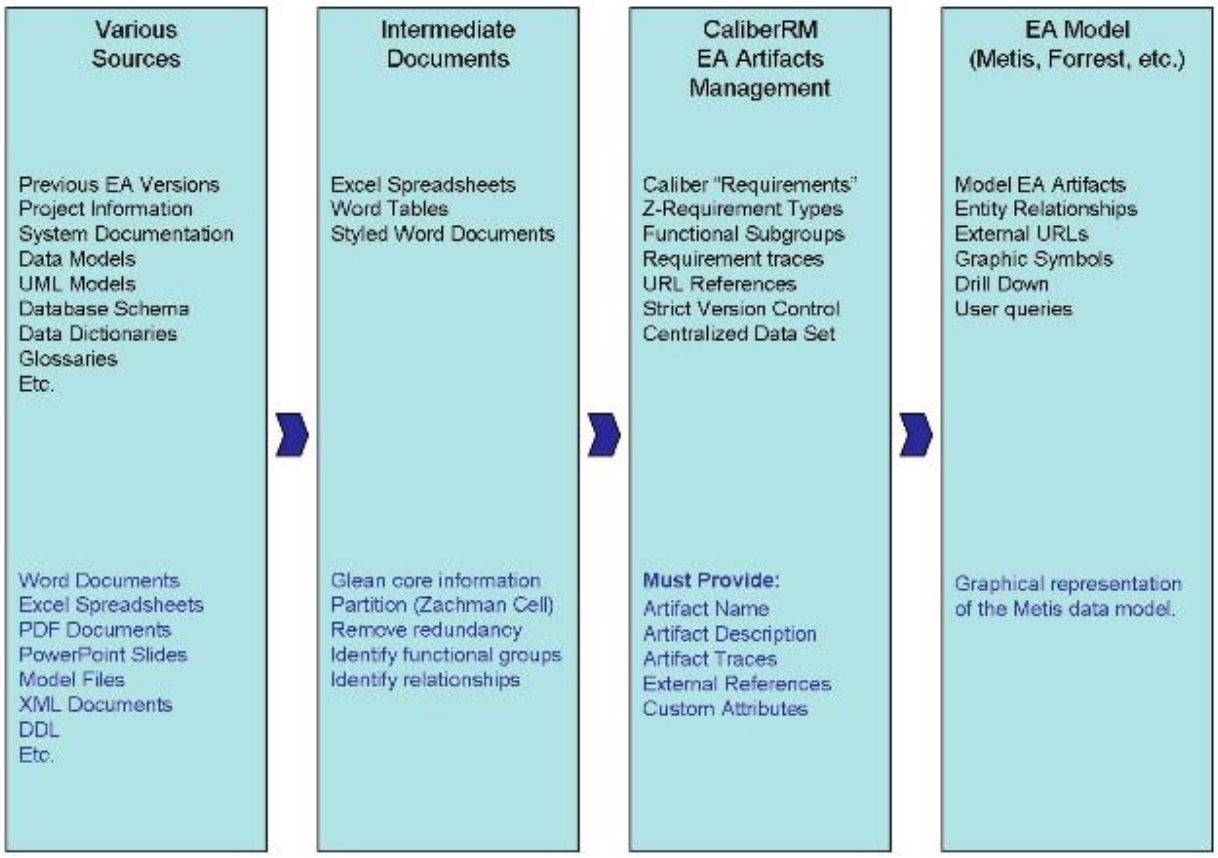

EA information is collected from various business and project source documents. The collected information is filtered for artifacts relevant to the VA Enterprise Architecture, normalized, and imported into the repository. The repository is being implemented as a centralized database server using the Borland CaliberRM. The data in the repository is structured according to the Zachman Framework and the primitive models that have been developed for each cell. Original source documents are stored on the centralized server so they can be referenced from either the EA repository or from the generated models (Metis) and web pages (Forrest). EA artifacts in the

repository are converted to a XML representation that is imported into Metis and Forrest. The EA has many potential VA users that will be served by the information generated for Metis and Forrest from CaliberRM XML data.

Additional information on the repository is in the following web pages: *EA Information Capture, Management, and Delivery*; *CaliberRM Mechanisms*; and, *Referencing External Mandates*.

## 1.2. EA Information Capture, Management, and Delivery

The VA Enterprise Architecture 3.0 (and later) will be represented by set of interrelated EA artifacts that reflect the real VA enterprise. An accurate and detailed representation of the enterprise will require the capture of thousands of artifacts, management of the resulting data set and the delivery of the managed artifacts to web-based graphical modeling tools and generated web pages. The following graphic presents some of the details of the required processes.



Document capture occurs in the two left-most stages. EA information is contained in a wide variety of source documents (mainly, but not exclusively, Word and PDF). Source documents include previous versions of the EA, external government mandates, internal VA policies and plans, and various project documents and databases.

The individual EA artifacts contained in these sources are captured in intermediate documents such as spreadsheets and tables. This phase involves the identification of EA artifacts, the partitioning of the information guided by the rules of the Zachman Framework, removal of redundant artifacts, and the identification of relationships between artifacts. The driving requirements for the capture process include:

1. Information capture must be complete.
2. Redundant information should be reduced.
3. Information should be partitioned and organized according to the rules of the Zachman Framework, as slightly modified by the VA Office of EA Management (OEAM).
4. The process must organize the Caliber information so that it directly supports the required visual models.
5. Relationships between artifacts should be retained (e.g., the relationship between things and activities).
6. The capture process must be easily reproducible.
7. The capture process must be efficient (although every artifact must be separately considered and verified).
8. The process must validate the final result to ensure that inaccuracies are minimized.

Captured EA artifacts are managed in a CaliberRM repository. This requirements/object-oriented database is perfect for managing individual artifacts. For example, CaliberRM requirements types are used classify artifacts according to the Zachman Framework. CaliberRM traces are used to relate artifacts as required. External reference capabilities are used to access external source documents. A complete list of how CaliberRM mechanisms are used to manage EA artifacts can be found under *CaliberRM Mechanisms*. Management also includes checking the artifacts and their relationships for quality (accuracy to the real world and internal consistency).

The EA artifacts are currently delivered to the user via the Metis modeling and Forrest web document tools. In fact, the EA approach is as open as technology will allow. The export from CaliberRM is packaged as XMI-standard XML. This XML stream is interfaced to Metis and Forrest to produce graphical and textual web pages. All the core data exists in the CaliberRM repository or in the interfaces between CaliberRM and Metis or Forrest. If requirements so dictate, it will be possible to replace these tools in the future and simply regenerate the output using the XML stream.

## 1.3. CaliberRM Mechanisms

Here is a summary of generic CaliberRM mechanisms that support EA modeling. The EA can use the capabilities represented by these mechanisms in most end-user reporting or modeling tools (via the internally developed XML interface). Some mechanisms did not make the list since we are not currently planning to make use of them.

To simplify this discussion, CaliberRM mechanisms are partitioned into the subsections below.

*Reference(s):*

- CaliberRM Mechanisms.pdf

### 1.3.1. Basic Mechanisms

Basic CaliberRM mechanisms represent indispensable capabilities required by every EA author.

1. **Requirement Types** – Document chapters and Zachman cells are organized by requirement type. This organization is useful since it is possible to create special properties to support different models in each cell. It is also possible to set security differently for each chapter or cell. Reporting by requirement type is much easier. In short, requirement types have made it possible to apply divide and conquer to a massive task.
2. **CaliberRM Requirements** – The basic information in a requirement is its name and description. This simple arrangement makes it possible to interpret a CaliberRM requirement as a generic enterprise architectural *object*. We have added an "object type" property to each object in order to support modeling tools that import CaliberRM XML data. Requirement descriptions retain original HTML formatting and image information that can be used by other modeling products.
3. **Nested Requirements** – CaliberRM leverages a parent-child relationship to construct a hierarchy of requirements within each requirement type. The current Metis XML interface transforms the context-sensitive hierarchical information encoded in CaliberRM to either Metis containers or Metis hierarchies.
4. **Internal Traceability** – CaliberRM provides the ability to trace from one object to another. For example, the architecture model contains traces from standard FEA functions to VA internal functions. This tracing puts VA activities into the wider Federal Government context without interfering with the VA-specific functional hierarchy. The tracings have direction (from or to the object) but no other properties (such as type, name, or function).
5. **External References** – CaliberRM offers three types of external references. (1) *Text references* are stored in CaliberRM. They provide a mechanism for comments on the object that need not appear in the description. (2) *File references* can reference files accessible from the client workstation. (3) *Web references* address objects via URL, making it possible to access them anywhere on the internet or intranet. EA 3.0 uses web references to maintain a controlled set of associated files on the CaliberRM server host.
6. **Shared Requirements** – Although EA 3.0 is normalizing its data, it is sometimes necessary

to reuse model objects. For example, some cells share the same taxonomy (the same object hierarchy). CaliberRM provides the capability to share the information between common objects. This means that the sharing object displays description information in the shared object, but cannot change it. Thus, EA 3.0 manages the descriptions of shared objects in only one place.

### 1.3.2. Utilities

The CaliberRM utilities are essential supplementary applications that support the authoring process.

1. **Glossaries** – CaliberRM provides the ability to import and manage project glossaries. The import uses a CaliberRM-specific XML-based glossary structure. Once in place, CaliberRM highlights and defines glossary words in the description on mouse-over. We need to determine if we can export the glossary to external models such as Metis or Borland OLAP (On-Line Analytical Processing). One curious issue is that glossary items are not case sensitive. This means that CaliberRM treats "of" as an acronym if "OF" (Optional Form) is in the glossary.
2. **Requirement Grid** – CaliberRM tends to view and update one requirement (object) at a time. The requirements grid is a handy utility to view and update requirements in bulk. Its minor query capability is used to filter requirements within a requirement type or a branch of the requirement hierarchy. Some of the displayed object parameters cannot be updated, but some can be bulk updated; most importantly, that includes requirement built-in and user-extended properties.
3. **Import from Microsoft Word** – CaliberRM provides a wizard to import requirements from Microsoft Word. Note that it is currently the *only* available import path. Although it does not allow updates, the wizard is simple to use and very effective. Word styles distinguish between requirement names and descriptions. CaliberRM treats header styles in a hierarchical fashion, making it possible to import complex hierarchies. The wizard offers a visual quality check before performing the actual import. Unfortunately, there is no way to import corresponding traces, external references, properties, etc. We are exploring, with Borland, the possibility of an XML-based import interface that might overcome these limitations.
4. **Export to Microsoft Access** – CaliberRM provides a wizard to export a project from its object-oriented database to an Access relational database. Most of the CaliberRM object-oriented database is unwound into a set of tables with primary and foreign key constraints. Oddly, the relational schema does not include a relationship between requirement type and the requirements that use that type. However, if we define unique requirement type tags, we can use that tag to select the desired requirement set. The Access database provides a lot of flexibility for database reports that are necessary to manage a large set of complex data. For example, we can produce requirement-level reports on properties, traces, and security, and we can use them to verify the data quality. External references and

version history information are not included in the export.

5. **External Traceability** – We plan to collect all essential enterprise data needed to describe the VA architecture. However, some detailed data is better left in the builders or managers format. Typical formats include Microsoft Project, Erwin, Oracle Designer, Rational Rose, etc. In an ideal world, we would be able to trace from CaliberRM objects *into* these "external" object sets. Currently, CaliberRM only provides external traceability via a Microsoft Project "plug-in". This will provide traceability from EA 3.0 events directly to Project schedule items.

### 1.3.3. Built-in Properties

Built-in CaliberRM properties add important specialized information to every EA artifact.

1. **System Attributes** – CaliberRM system attributes are user-extensible. Two of these are relevant to EA 3.0:
    1. **Requirement Status** – Factory settings are Submitted, Pending, Accepted, Draft, and Deferred. EA 3.0 initial input defaults to Submitted. We can add additional status choices as required.
    2. **Requirement Priority** – Factory settings are Essential, Useful, Desirable, and Unassigned. EA 3.0 initial input defaults to Unassigned. We can add additional priorities as required.
2. **Requirement Owner** – Caliber requirement (object) owners are the only persons that can update the requirement. The CaliberRM administrator can assign requirement ownership to a group of persons.
3. **Requirement Version** – CaliberRM faithfully records every requirement (object) change as an internal version. It is possible select a previous version so you can see the change. In addition, CaliberRM will provide a complete report for any requirement with a previous version.
4. **Requirement Tag** – The CaliberRM tag uniquely identifies a requirement type. CaliberRM combines the Tag with a sequence number to create a unique requirement tag for every requirement (object). CaliberRM also maintains a unique internal requirement identifier that the EA 3.0 XML interface uses to tag objects for the Metis model.

### 1.3.4. User-Extended Properties

The CaliberRM mechanisms provide for additional user-extended properties. These extensions are essential for encoding EA modeling information in every EA artifact.

1. **Internal Only** – We maintain this flag to indicate that a requirement (object) is not ready for general use. For example, if this binary flag is on (the default), the object will not be included in the current model.
2. **Procurement Sensitive** – Even if the Internal Only flag is off, only selected VA staff will be able to see an object if is marked procurement sensitive.

3. **Internal Version** – This number is the working version of the architecture.
4. **Source** – We track the original source for the object using a single-selection drop-down list. As we acquire new sources, we add them to the list. It is an important step in the bulk load process to update this property.
5. **Object Type** – We assign the object type from a single-selection drop-down list according to the EA 3.0 Zachman Framework-based model. The XML interface transmits the object type to the Metis model where it labels the model object or provides an appropriate icon.

### 1.3.5. Data Management

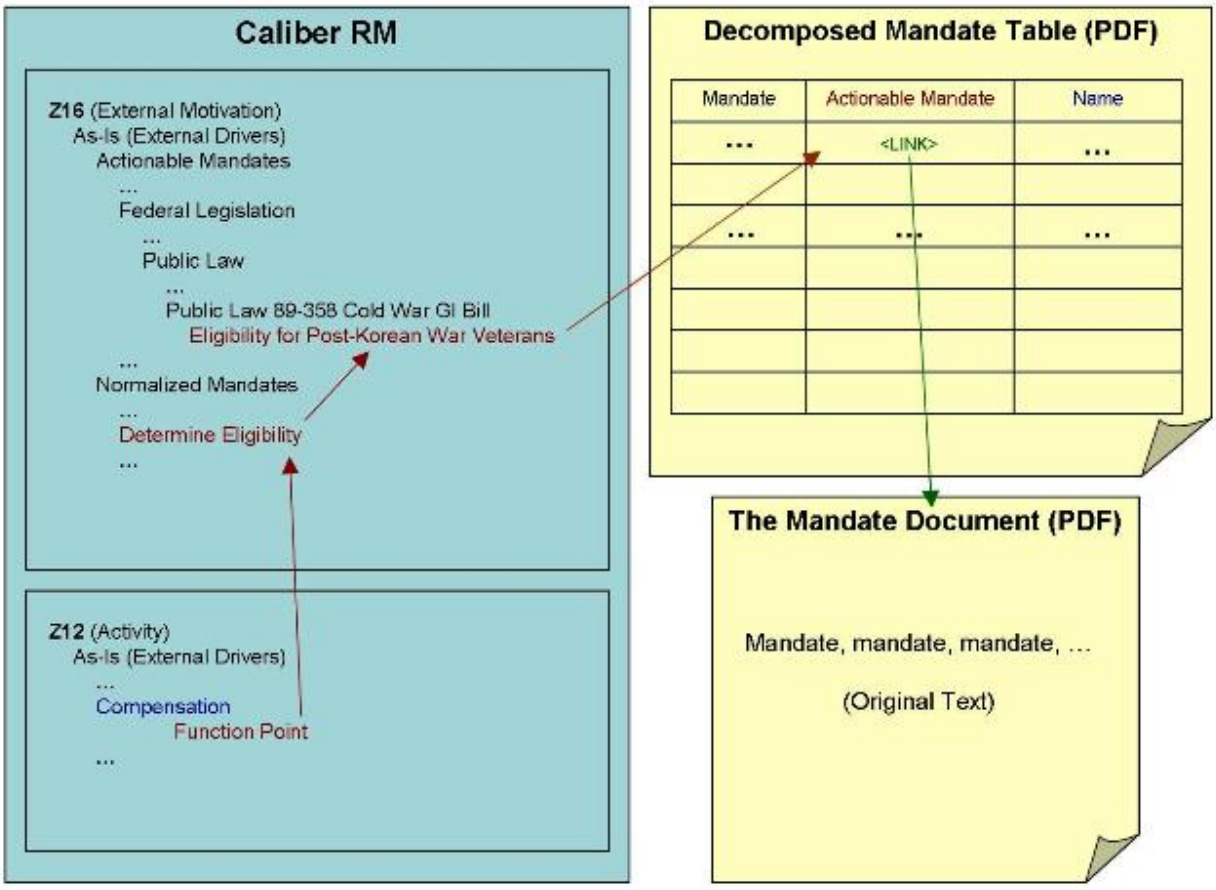
The trusted, efficient management of a large set of EA artifacts is the most important requirement satisfied by CaliberRM. The underlying object-oriented database provides critical mechanisms that satisfy this requirement.

1. **Requirement Locking** – CaliberRM runs on a centralized server. Proprietary CaliberRM clients access the server using proprietary application protocols over the network. The multi-user server uses a locking mechanism to avoid race conditions on a requirement. When a user starts an update of any requirement information (name, description, property, trace … whatever) the requirement is locked against updates by others. As soon as the user saves the update, it is made available for access by others. If they are out-of-synch, CaliberRM refreshes the view to reflect the update. All in all, a tidy way to manage the data.
2. **Requirement History** – CaliberRM maintains a history of every change made to a requirement. CaliberRM provides a simple interface to peruse the changes, right back to original object creation.
3. **Requirement Security** – CaliberRM provides management of user responsibilities and project groups. This feature will become increasing important as EA project collaboration with VA administrations increases. The project administrator uses a separate (restricted) application (the Framework Administrator) to manage CaliberRM users.
4. **Requirement Discussions** – CaliberRM provides a special tab to support data management collaboration for each requirement (object). Users can post new comments or respond to old ones. CaliberRM maintains and displays the discussion history in a hierarchical list.
5. **Database Backup** – The CaliberRM database objects are in proprietary files that the project administrator can periodically back up or restore as required.

## 1.4. Referencing External Mandates

The Enterprise Architecture models the motivation that drives the Department of Veterans Affairs in Column 6 of the Zachman Framework. The motivation exists as a set of documents external to the architecture; these include mandates (laws, orders, etc.), strategic plans, department policies, project plans, application designs, deployment plans, system guides, and so on. This note describes how we identify and include the actionable mandates in the architecture. It also provides an example of how the actionable mandates provide VA's business requirements.

The example below illustrates how the Enterprise Architecture will link to the external mandate documents required to support the models developed for Zachman Framework Row 1, Column 6 (Cell Z16).




In order to model Z16, it will be necessary to analyze hundreds of external Federal and State laws and policies, executive orders, local ordinances, regulations, the constitution, and even common law. However, only certain sections, paragraphs, and sentences of these mandates require action by Veterans Affairs. Those sections must be decomposed (identified, parsed, and paraphrased) as "actionable mandates". Analysts will accomplish this by creating a word table that contains every header and paragraph in the original document. They manually scan the decomposed table for VA actionable mandates, which they name and paraphrase. Then, the actionable mandates will be bulk loaded into CaliberRM where they will be available for

classification according to a predetermined taxonomy. The analyst further organizes the actionable mandates by creating “normalized mandates” that group similar actionable mandates (perhaps from different mandate documents) for use in the architecture as external business requirements. Finally, for example, the analyst identifies the business function (found in Zachman cell Z12) motivated by the normalized actionable mandate and creates a “function point” that connects function and mandate.

An automated process will export this organized information, managed in the CaliberRM repository, to a graphical model based on the Zachman Framework. Users will be able to scan and query the information model (implemented with the Metis modeling tool in the first version of this architecture). They will also be able to link from the actionable mandate to the decomposed mandate document. Finally, they will be able to link from the decomposed mandate to the original document, when original context is required.

The following table is an example of a decomposed mandate with identified actionable mandates.

| **PUBLIC LAW 104-191** | **Actionable Mandate** | **Name** |
|---|---|---|
| "(1) A health plan. | | |
| "(2) A health care clearinghouse. | | |
| "(3) A health care provider who transmits any health information in electronic form in connection with a transaction referred to in section 1173(a)(1). | | |
| "(b) REDUCTION OF COSTS.--Any standard adopted under this part shall be consistent with the objective of reducing the administrative costs of providing and paying for health care. | Any standard adopted under Section 1172 shall be consistent with the objective of reducing the administrative costs of providing and paying for health care. | Section 1172 Reduce Costs |
| "(c) ROLE OF STANDARD SETTING ORGANIZATIONS.-- | | |
| "(1) IN GENERAL.--Except as provided in paragraph (2), any | Adopt only those data transmission standards set by | Section 1172 Adopt Specific Organization Standards |

| | | |
|---|---|---|
| standard adopted under this part shall be a standard that has been developed, adopted, or modified by a standard setting organization. | standard-setting organizations | |
| "(2) SPECIAL RULES.-- | | |
| "(A) DIFFERENT STANDARDS.--The Secretary may adopt a standard that is different from any standard developed, adopted, or modified by a standard setting organization, if-- | | |
| "(i) the different standard will substantially reduce administrative costs to health care providers and health plans compared to the alternatives; and | | |
| "(ii) the standard is promulgated in accordance with the rulemaking procedures of subchapter III of chapter 5 of title 5, United States Code. | | |
| "(B) NO STANDARD BY STANDARD SETTING ORGANIZATION.--If no standard setting organization has developed, adopted, or modified any standard relating to a standard that the Secretary is authorized or required to adopt under this part-- | The Secretary should approve the use of any data transmission standard that was not developed by a standard-setting organization | Section 1172 Obtain Secretary Approval |
| "(i) paragraph (1) shall not apply; and | | |
| "(ii) subsection (f) shall apply. | | |
| (3) CONSULTATION REQUIREMENT.-- | | |